Option B : Examen du 5 Janvier

Splines cubiques

Il est important de rédiger de façon claire **ET** synthétique.
On peut bien entendu admettre le résultat de n'importe quelle question pour passer à la suivante.

PARTIE THÉORIQUE

1 A propos de l'interpolation de Hermite

On se donne deux points distincts $x_g < x_d$ ainsi que quatre réels y_g, y_d, z_g et z_d.

1. Montrer que l'application linéaire

$$\Phi : \mathbb{R}_3[X] \mapsto \begin{pmatrix} p(x_g) \\ p'(x_g) \\ p(x_d) \\ p'(x_d) \end{pmatrix},$$

   est bijective, où on rappelle que $\mathbb{R}_3[X]$ est l'ensemble des polynômes à coefficients réels de degré inférieur ou égal à 3.
   En déduire qu'il existe un unique polynôme de degré inférieur ou égal à 3 vérifiant

$$\begin{cases} p(x_g) = y_g, \\ p(x_d) = y_d, \\ p'(x_g) = z_g, \\ p'(x_d) = z_d. \end{cases}$$

   Ce polynôme est appelé le polynôme d'interpolation de Hermite pour ces données.

2. Donner une expression explicite de p en fonction des données. On pourra chercher p sous la forme

$$p(x) = q_0(x) + (x - x_g)(x - x_d)q_1(x),$$

   où q_0 et q_1 sont des polynômes de degré inférieur ou égal à 1 que l'on déterminera, par exemple, dans la base des polynômes $x - x_g$ et $x - x_d$.

3. Montrer qu'il existe une constante universelle $C > 0$ (i.e. indépendante des $x_\bullet, y_\bullet$ et $z_\bullet$) telle qu'on ait

$$\sup_{x\in[x_g,x_d]} |p(x)| \leq C(|y_g| + |y_d|) + C|x_d - x_g|(|z_g| + |z_d|).$$

4. Calculer $p''(x_g)$ et $p''(x_d)$ en fonction de $x_g, x_d, y_g, y_d, z_g, z_d$.

5. Soit maintenant f une fonction de classe $\mathcal{C}^4$. On suppose que les données d'interpolation sont définies par

$$y_g = f(x_g), y_d = f(x_d), z_g = f'(x_g), \text{ et } z_d = f'(x_d).$$

   Montrer que dans ces conditions, le polynôme de Hermite p vérifie l'estimation d'erreur suivante

$$\forall x \in [x_g, x_d], \exists \xi \in]x_g, x_d[, \text{ tel que } f(x) - p(x) = (x - x_g)^2(x - x_d)^2 \frac{f^{(4)}(\xi)}{4!}.$$

6. En déduire que

$$\sup_{[x_g,x_d]} |f - p| \leq |x_d - x_g|^4 \frac{\|f^{(4)}\|_\infty}{384}.$$

2 Existence et unicité de la spline cubique contrainte interpolante

On se donne des points $x_1 < ... < x_n$ équi-distants ainsi que n valeurs $y_1, ..., y_n$, et deux autres réels z_d et z_n. On notera h le pas de la subdivision, i.e. $x_{i+1} - x_i = h$ pour tout i.

Le but de cette partie est de démontrer le résultat suivant : il existe une unique fonction $\pi : [x_1, x_n] \to \mathbb{R}$ vérifiant

$$\pi \text{ est de classe } \mathcal{C}^2 \text{ sur } [x_1, x_n], \qquad (\mathcal{P}_1)$$

$$\pi \text{ interpole les valeurs données, i.e. } \pi(x_i) = y_i, \ \ \forall i = 1, ..., n. \qquad (\mathcal{P}_2)$$

La restriction de π à chaque intervalle $[x_i, x_{i+1}]$ est un polynôme de degré inférieur ou égal à 3. $(\mathcal{P}_3)$

$$\pi'(x_1) = z_1, \text{ et } \pi'(x_n) = z_n. \qquad (\mathcal{P}_4)$$

Une fonction vérifiant $(\mathcal{P}_1)$,$(\mathcal{P}_2)$ et $(\mathcal{P}_3)$ est appelée une spline cubique interpolante. Si de plus, elle vérifie une condition du type $(\mathcal{P}_4)$, elle est dite **contrainte**.

1. Montrer qu'une telle fonction π est complètement déterminée par la connaissance des $(\pi(x_i))_{1\leq i\leq n}$ et des $(\pi'(x_i))_{1\leq i\leq n}$.

   Par hypothèse, les valeurs de π aux noeuds sont connues, on va donc chercher dans la suite à déterminer les valeurs que π' doit prendre en ces noeuds pour vérifier toutes les conditions demandées.

2. Soient donc $z_1, z_2, ..., z_{n-1}, z_n$ les valeurs de π' aux noeuds à déterminer (la première et la dernière sont en fait des données du problème).

   Ecrire les équations (dépendant de h) que doivent vérifier les $(y_i)_i$ et les $(z_i)_i$ pour que la fonction π associée soit de classe $\mathcal{C}^2$.

3. Ecrire ces équations sous la forme d'un système linéaire carré de taille $n-2$ noté $AZ = G$ où $Z = (z_2, ..., z_{n-1})$ et où G est un second membre dépendant des données $y_1, ..., y_n$ et z_1, z_n que l'on explicitera.

4. Montrer que si ce système admet une solution alors on a l'estimation

$$\|Z\|_\infty = \max_{2\leq i\leq n-1} |z_i| \leq \frac{|z_1|}{2} + \frac{|z_n|}{2} + 3 \max_{2\leq i\leq n-1} \left| \frac{y_{i+1} - y_{i-1}}{2h} \right|.$$

   On pourra s'intéresser à un indice $2 \leq i \leq n-1$ pour lequel $|z_i| = \|Z\|_\infty$, et regarder l'équation correspondante du système $AZ = G$.

5. En déduire que le problème $AZ = G$ admet une unique solution et que celle-ci vérifie

$$\|Z\|_\infty \leq 4\|f'\|_{L^\infty(]a,b[)}.$$

   Conclure enfin à l'existence et unicité de la spline cubique contrainte interpolante pour f.

PARTIE NUMÉRIQUE

3 Mise en oeuvre et expérimentations numériques

On reprend ici les notations précédentes. On se donne un intervalle $[a, b]$ de $\mathbb{R}$ et une fonction f de classe $\mathcal{C}^4$ sur $[a, b]$. On suppose que la discrétisation est choisie de sorte que $x_1 = a$ et $x_n = b$, et donc que

$$h = \frac{b-a}{n-1}.$$

On pose maintenant

$$y_i = f(x_i), \ \ \forall i \in \{1, ..., n\}, \text{ et } z_1 = f'(x_1), \ \ z_n = f'(x_n).$$

On souhaite calculer en Scilab la spline cubique interpolante π correspondant à ces données et évaluer numériquement l'erreur d'interpolation entre f et π (resp. en f' et π') en fonction du pas de la discrétisation.

Plus précisément, on admet qu'on peut démontrer (**ce n'est pas demandé ici**) que l'on a les estimations suivantes (pour un $C > 0$ indépendant de h, donc de n)

$$\max_{1 \leq i \leq n} |f'(x_i) - \pi'(x_i)| \leq Ch^4, \tag{E_1}$$

$$\|f' - \pi'\|_{L^\infty(]a,b[)} \leq Ch^3, \tag{E_2}$$

$$\|f - \pi\|_{L^\infty(]a,b[)} \leq Ch^4. \tag{E_3}$$

1. Ecrire une fonction Scilab

```
function [Y,Z]= parametres_spline (a,b,n,f,f_prime)
   ....
endfunction
```

qui, à partir de la donnée de a, b, f, f' et n calcule les valeurs $(\pi(x_i))_{1 \leq i \leq n}$ et $(\pi'(x_i))_{1 \leq i \leq n}$ de la spline cubique interpolante π et de sa dérivée aux points d'interpolation équirépartis sur $[a, b]$. On utilisera la méthode décrite dans la question II.3.

2. Illustrer numériquement sur un exemple, à l'aide d'une courbe en échelle logarithmique, l'estimation (E_1). On pourra par exemple choisir $a = 0, b = 1$ et $f(x) = \sin(10x + 1)$, mais tout autre exemple (non trivial) peut tout à fait convenir.

3. Ecrire une fonction Scilab

```
function [i]= indice (a,b,n,x)
   ....
endfunction
```

qui pour a, b, n et $x \in]a, b[$ donnés, calcule l'unique indice $i \in \{1, ..., n-1\}$ tel que $x \in [x_i, x_{i+1}[$.

4. Ecrire deux fonctions Scilab

```
function [val_pi]= valeur_spline (a,b,n,Y,Z,x)
   ....
endfunction

function [val_pi_prime]= valeur_derivee_spline (a,b,n,Y,Z,x)
   ....
endfunction
```

qui pour a, b, n, Y, Z et $x \in]a, b[$ donnés, calculent respectivement les valeurs $\pi(x)$ et $\pi'(x)$ de la spline cubique pour les données d'interpolation $Y = (y_1, ..., y_n)$ et $Z = (z_1, ..., z_n)$ et de sa dérivée au point x.

5. Pour tester numériquement les estimations (E_2), (E_3) on est confrontés à la difficulté de l'estimation de la norme L^∞ de la différence de deux fonctions. Pour ce faire on utilise la méthode aléatoire suivante
   — Choisir un entier $M > 0$ suffisamment grand.

- Tirer aléatoirement [1] M points choisis selon la loi uniforme sur $[a, b]$ et notés $\bar{x}^k$, $k = 1, ..., M$ (ne pas confondre avec les points d'interpolation $x_1, ..., x_n$).
- On décide alors d'estimer la norme infinie d'une fonction g par

$$\|g\|_\infty \sim \max_{1 \leq k \leq M} |g(\bar{x}^k)|.$$

Programmer deux fonctions Scilab

```
function [erreur]= erreur_spline (a,b,f,f_prime,n,M)
   ....
endfunction

function [erreur]= erreur_derivee_spline (a,b,f,f_prime,n,M)
   ....
endfunction
```

qui pour a, b, f, f', n et M donnés, renvoient respectivement une estimation de $\|f - \pi\|_\infty$ et de $\|f' - \pi'\|_\infty$ calculées par la méthode précédente.
Illustrer à l'aide de courbes logarithmiques les estimations d'erreur (E_2),(E_3). On fixera par exemple $M = 5000$, ce qui suffira pour des valeurs de n inférieures à 100 par exemple.

1. La commande Scilab `a+(b-a)*rand([1:M])` permet de faire cela

Corrigé

1 A propos de l'interpolation de Hermite

1. Comme $\mathbb{R}_3[X]$ et $\mathbb{R}^4$ ont la même dimension, et comme Φ est linéaire, il suffit de vérifier que son noyau est réduit à 0. Or, si un polynôme $p \in \mathbb{R}_3[X]$ est dans le noyau de Φ, cela signifie que
$$p(x_g) = p'(x_g) = 0,$$
$$p(x_d) = p'(x_d) = 0.$$
Ceci montre que x_g et x_d sont deux racines doubles distinctes de p, donc le polynôme $(x - x_g)^2(x - x_d)^2$ doit diviser p. Comme $\deg(p) \leq 3$, ceci n'est possible que si p est nul, ce qui conclut la preuve.

2. On adopte l'écriture proposée
$$p(x) = q_0(x) + (x - x_g)(x - x_d)q_1(x),$$
avec
$$q_0(x) = a(x - x_g) + b(x - x_d),$$
$$q_1(x) = \alpha(x - x_g) + \beta(x - x_d).$$
En évaluant p en x_g et x_d, on trouve de suite
$$p(x_g) = b(x_g - x_d), \text{ et } p(x_d) = a(x_d - x_g),$$
ce qui donne
$$b = \frac{y_g}{x_g - x_d}, \text{ et } a = \frac{y_d}{x_d - x_g}.$$
On évalue maintenant p' en x_g et x_d et on trouve
$$p'(x_g) = (a + b) + (x_g - x_d)q_1(x_g) = (a + b) + \beta(x_d - x_g)^2,$$
$$p'(x_d) = (a + b) + (x_d - x_g)q_1(x_d) = (a + b) + \alpha(x_d - x_g)^2.$$
Ce qui donne
$$\beta = \frac{1}{(x_d - x_g)^2}\left(z_g - \frac{y_d - y_g}{x_d - x_g}\right),$$
$$\alpha = \frac{1}{(x_d - x_g)^2}\left(z_d - \frac{y_d - y_g}{x_d - x_g}\right).$$
L'expression du polynôme de Hermite est donc in fine
$$p(x) = \frac{y_d}{x_d - x_g}(x - x_g) + \frac{-y_g}{x_d - x_g}(x - x_d) + \frac{(x - x_g)(x - x_d)}{(x_d - x_g)^2}\left[\left(z_d - z_{g/d}\right)(x - x_g) + \left(z_g - z_{g/d}\right)(x - x_d)\right],$$
où on a défini
$$z_{g/d} = \frac{y_d - y_g}{x_d - x_g}.$$
Pour la partie numérique, on aura également besoin de l'expression de la dérivée du polynome de Hermite
$$p'(x) = z_{g/d} + \frac{(x - x_g)^2}{(x_d - x_g)^2}(z_d - z_{g/d}) + \frac{(x - x_d)^2}{(x_d - x_g)^2}(z_g - z_{g/d}) + 2\frac{(x - x_g)(x - x_d)}{(x_d - x_g)^2}(z_g + z_d - 2z_{g/d}).$$

3. Pour tout $x \in [x_g, x_d]$, en reprenant les notations précédentes, nous avons
$$\begin{aligned}|p(x)| &\leq |a||x - x_g| + |b||x - x_d| + |x - x_g||x - x_d|(|\alpha||x - x_g| + |\beta||x - x_d|)\\ &\leq |y_g| + |y_d| + |x_d - x_g|^2\left(\frac{|z_g||x_d - x_g| + |y_g| + |y_d|}{|x_d - x_g|^2} + \frac{|z_d||x_d - x_g| + |y_g| + |y_d|}{|x_d - x_g|^2}\right)\\ &\leq 3(|y_g| + |y_d|) + |x_d - x_g|(|z_g| + |z_d|).\end{aligned}$$
Ainsi, $C = 3$ convient par exemple (ce n'est probablement pas la constante optimale d'ailleurs, mais ça n'a aucune importance ...)

4. On repart de la formule de p suivante
$$\begin{aligned} p(x) &= a(x-x_g) + b(x-x_d) + (x-x_g)(x-x_d)\big(\alpha(x-x_g)+\beta(x-x_d)\big) \\ &= a(x-x_g) + b(x-x_d) + \alpha(x-x_g)^2(x-x_d) + \beta(x-x_g)(x-x_d)^2. \end{aligned}$$
On veut calculer $p''(x_g)$, il faut donc réfléchir aux termes qui vont nécessairement s'annuler et ne pas calculer $p''(x)$ puis remplacer $x = x_g$ sous peine d'obtenir des calculs un peu plus lourds et d'augmenter donc les risques d'erreur.
On va donc utiliser les formules suivantes que je vous invite à vérifier
$$\begin{aligned} \frac{d^2}{dx^2}\bigg((x-x_g)^2 f(x)\bigg)_{|x=x_g} &= 2f(x_g), \\ \frac{d^2}{dx^2}\bigg((x-x_g)g(x)\bigg)_{|x=x_g} &= 2g'(x_g). \end{aligned}$$
On obtient donc
$$p''(x_g) = 2\alpha(x_g - x_d) + 4\beta(x_g - x_d),$$
et de la même façon
$$p''(x_d) = 4\alpha(x_d - x_g) + 2\beta(x_d - x_g)$$
En utilisant les expressions établies plus haut de α et β en fonction des données du problème, cela donne
$$\begin{aligned} p''(x_g) &= 2(\alpha+2\beta)(x_g - x_d) = \frac{-2}{x_d - x_g}\left(2z_g + z_d - 3\frac{y_1 - y_g}{x_d - x_g}\right), \\ p''(x_d) &= 2(2\alpha+\beta)(x_d - x_g) = \frac{2}{x_d - x_g}\left(z_g + 2z_d - 3\frac{y_1 - y_g}{x_d - x_g}\right). \end{aligned}$$

5. On utilise ici la méthode usuelle. On fixe un $x \in [x_g, x_d]$ et on suppose que x n'est ni égal à x_g, ni à x_d, sinon il n'y a rien à démontrer (toute valeur de ξ convient).
On introduit alors le nombre M_x défini par l'égalité
$$f(x) - p(x) = \frac{M_x}{4!}(x-x_g)^2(x-x_d)^2,$$
puis on considère la fonction
$$g(y) = f(y) - p(y) - \frac{M_x}{4!}(y-x_g)^2(y-x_d)^2.$$
Par construction (i.e. par choix de M_x !), g s'annule en x mais on a aussi
$$\begin{aligned} g(x_g) &= f(x_g) - p(x_g) = 0, \\ g'(x_g) &= f'(x_g) - p'(x_g) = 0, \\ g(x_d) &= f(x_d) - p(x_d) = 0, \\ g'(x_d) &= f'(x_d) - p'(x_d) = 0. \end{aligned}$$
Ainsi, x_g et x_d sont des racines doubles de g. Au total, en comptant les multiplicités des racines, g s'annule donc 5 fois. D'après le théorème de Rolle généralisé, on sait qu'il existe un $\xi \in]x_g, x_d[$ en lequel la dérivée quatrième de g s'annule. Comme p est de degré au plus 3, il ne contribue pas au calcul de la dérivée quatrième de g et on obtient donc, tous calculs faits, que
$$M_x = f^{(4)}(\xi),$$
ce qu'il fallait démontrer.

6. D'après la question précédente, on a
$$\sup_{[x_g,x_d]} |f-p| \leq \frac{\|f^{(4)}\|_\infty}{4!} \sup_{[x_g,x_d]} (|x-x_g|^2|x-x_d|^2),$$
et il reste à calculer le sup dans le membre de droite. On constate qu'il est aussi égal à
$$\left(\sup_{[x_g,x_d]} |x-x_g||x-x_d|\right)^2,$$
qui est un sup atteint en $x = (x_g + x_d)/2$ et qui vaut donc $(1/4)^2$. En remplaçant dans la formule ci-dessus, on trouve le résultat annoncé.

2 Existence et unicité de la spline cubique contrainte interpolante

1. La restriction de π sur chaque intervalle $[x_i, x_{i+1}]$ est un polynôme de degré 3. On a vu dans la première partie qu'un tel polynôme est déterminé de manière unique par les valeurs π et de sa dérivée en x_i et x_{i+1}.
2. On utilise les formules établies dans la première partie qui donnent les valeurs des dérivées secondes du polynôme de Hermite en fonction des données. Etant donnés les y_i et les z_i, on peut donc calculer les dérivées secondes à gauche et à droite en chaque noeud x_i, $i = 2, ..., n-1$ de la façon suivante
$$\pi''(x_i^-) = \frac{2}{h}\left(z_{i-1} + 2z_i - 3\frac{y_i - y_{i-1}}{h}\right),$$
$$\pi''(x_i^+) = \frac{-2}{h}\left(2z_i + z_{i+1} - 3\frac{y_{i+1} - y_i}{h}\right).$$
Ainsi, pour que la fonction π obtenue en recollant les différents polynômes de Hermite sur chaque intervalle soit de classe $\mathcal{C}^2$, il faut et il suffit que les équations $\pi''(x_i^-) = \pi''(x_i^+)$ pour $i = 2, ..., n-1$, soient vérifiées, ce qui donne
$$\frac{2}{h}\left(z_{i-1} + 2z_i - 3\frac{y_i - y_{i-1}}{h}\right) = \frac{-2}{h}\left(2z_i + z_{i+1} - 3\frac{y_{i+1} - y_i}{h}\right).$$
3. On obtient la forme annoncée en introduisant les matrices et vecteurs de taille $n-2$ suivants
$$A = \frac{1}{h}\begin{pmatrix} 8 & 2 & 0 & \cdots & \cdots & 0 \\ 2 & 8 & 2 & \ddots & & \vdots \\ 0 & 2 & 8 & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 2 & 8 & 2 \\ 0 & \cdots & \cdots & 0 & 2 & 8 \end{pmatrix},$$
et
$$G = \begin{pmatrix} 6\frac{y_3 - y_1}{h^2} - \frac{2z_1}{h} \\ 6\frac{y_4 - y_2}{h^2} \\ \vdots \\ 6\frac{y_{n-1} - y_{n-3}}{h^2} \\ 6\frac{y_n - y_{n-2}}{h^2} - \frac{2z_n}{h} \end{pmatrix}.$$
4. On suit l'indication de l'énoncé. On choisit $2 \le i \le n-1$ tel que $|z_i| = \|Z\|_\infty$.
   — On suppose d'abord que $3 \le i \le n-2$. La i-ième équation du système s'écrit
   $$\frac{1}{h}(2z_{i-1} + 8z_i + 2z_{i+1}) = \frac{6}{h^2}(y_{i+1} - y_{i-1}),$$
   et donc
   $$\frac{8}{h}z_i = \frac{6}{h^2}(y_{i+1} - y_{i-1}) - \frac{2z_{i-1} + 2z_{i+1}}{h},$$
   et en prenant la valeur absolue
   $$\frac{8}{h}|z_i| \le \frac{6}{h^2}(y_{i+1} - y_{i-1}) + \frac{2}{h}(|z_{i-1}| + |z_{i+1}|) \le \frac{6}{h^2}|y_{i+1} - y_{i-1}| + \frac{4}{h}\|Z\|_\infty,$$
   et comme on a choisi i pour que $|z_i| = \|Z\|_\infty$, on déduit
   $$\|Z\|_\infty \le \frac{3}{2h}|y_{i+1} - y_{i-1}|,$$
   ce qui implique le résultat annoncé.

— Si $i=2$, l'équation correspondante du système devient

$$\frac{1}{h}(8z_2+2z_3)=\frac{6}{h^2}(y_3-y_1)-\frac{2z_1}{h},$$

et donc

$$\frac{8}{h}|z_2|=\frac{6}{h^2}|y_3-y_1|+\frac{2|z_3|}{h}+\frac{2|z_1|}{h}.$$

Comme $|z_3|\leq \|Z\|_\infty$ [2], on a donc dans ce cas

$$\frac{6}{h}\|Z\|_\infty\leq \frac{6}{h^2}|y_3-y_1|+\frac{2|z_1|}{h},$$

et enfin

$$\|Z\|_\infty\leq 2\frac{|y_3-y_1|}{2h}+\frac{|z_1|}{3}.$$

Ceci implique aussi l'inégalité attendue.

— Le cas $i=n-1$ se traite exactement de la même façon.

5. Il suffit de remarquer que la matrice A est carrée et injective. En effet, d'après la question précédente, si on prend $z_1=z_n=0$ et $y_1=....=y_n=0$, alors le second membre G est nul et toute solution de $AZ=0$ vérifie l'inégalité de la question précédente qui nous dit que $\|Z\|_\infty=0$ et donc $Z=0$.
Ainsi, le vecteur Z est parfaitement défini par la donnée des y_i et de z_1, z_n. De plus, la définition de ces données nous donne

$$|z_1|=|f'(x_1)|\leq\|f'\|_\infty,$$
$$|z_n|=|f'(x_n)|\leq\|f'\|_\infty,$$
$$\frac{|y_{i+1}-y_{i-1}|}{2h}=\frac{|f(x_{i+1})-f(x_{i-1})|}{|x_{i+1}-x_{i-1}|}\leq\|f'\|_\infty.$$

L'inégalité de la question 4 nous donne donc bien

$$\|Z\|_\infty\leq\|f'\|_\infty.$$

Une fois que $Z=(z_2,...,z_{n-1})$ est déterminé, la question 1 nous montre que π est parfaitement déterminé car on connaît toutes les valeurs de π et π' aux noeuds x_i de la discrétisation.

2. mais on a pas $|z_1|\leq\|Z\|_\infty$ car z_1 est une donnée qui n'apparaît pas dans le vecteur des inconnues $Z=(z_2,...,z_{n-1})$

3 Mise en oeuvre et expérimentations numériques

1. Voici une façon de procéder pour le calcul de la spline, il s'agit d'assembler le système linéaire $AZ = G$ introduit dans la partie précédente puis de compléter le vecteur résultat par les deux valeurs de la dérivée de f (connues) aux deux bornes de l'intervalle.

```
function [Y,Z]= parametres_spline (a,b,n,f,f_prime)

   x=linspace(a,b,n)';  // Les points d'interpolation en colonne
   Y=f(x);              // Valeurs de f aux points d'interpolation
                        // Par définition ce seront aussi les valeurs de pi

   h=(b-a)/(n-1);       // Pas de la discrétisation

   // Construction par diagonales de la matrice A

   A=4*diag(ones(n-2,1)) + 2*diag(ones(n-3,1),1);
   A=A+A';
   A=(1/h)*A;

   // Construction du second membre (attention aux indices !)

   G=(6/h^2) * (Y(3:$)- Y(1:$-2));        // La partie en y
   G(1)=G(1) - (2/h)*f_prime(x(1));       // Le terme en z1
   G(n-2)=G(n-2) - (2/h) * f_prime(x(n)); // Le terme en zn


   // On résout le système (de taille n-2 !)

   Z=A\G;          // Ceci nous donne les dérivées de la spline
                   // en dehors des points du bord

   Z=[f_prime(x(1)); Z; f_prime(x(n))];  // On ajoute les valeurs au bord

endfunction
```

2. Voici le programme qu'on propose pour traiter le cas de l'énoncé (à mettre juste après la définition de la fonction `parametres_spline` bien sûr !).
   On observe que l'erreur se comporte bien en h^4 comme prévu par la théorie.

```
a=0;
b=1;
deff('y=f(x)','y=sin(10*x+1)');
deff('y=f_prime(x)','y=10*cos(10*x+1)');


erreurs=[];               // Initialisation d'un tableau vide

for n=10:20:200           // Boucle sur le nombre de pas
  x=linspace(a,b,n)';     // En colonne
  [Y,Z]= parametres_spline (a,b,n,f,f_prime);     // Calcul de la spline
  erreurs=[erreurs; ...
           (b-a)/(n-1)  max(abs(Z-f_prime(x)))]; // Stockage du résultat
                                                 // Dans une nouvelle ligne
                                                 // qui contient le pas et l'erreur
end;

// On trace l'erreur en fonction du pas
plot(erreurs(:,1), erreurs(:,2),'-*');

// On trace h --> h^4 en rouge pour comparer les pentes
plot(erreurs(:,1), erreurs(:,1).^4,'r');

h=gca();
h.log_flags='lln';  // Echelle logarithmique
```

3. Il suffit d'utiliser à bon escient la commande `floor`.

```
function [i]= indice (a,b,n,x)

   h=(b-a)/(n-1);            // Pas de la discrétisation

   i = floor((x-a)/h)+1;     // La commande floor retourne l'entier
                             // immédiatement inférieur

endfunction
```

4. On utilise juste les formules explicites du polynôme de Hermite et de sa dérivée données dans le corrigé de la première partie, une fois qu'on a trouvé l'intervalle de travail $[x_i, x_{i+1}]$ correspondant à la valeur x que l'on veut calculer, ce qui est fait par la fonction `indice`.

```
function [val_pi]= valeur_spline (a,b,n,Y,Z,x)

   h=(b-a)/(n-1);            // Pas de la discrétisation

   i=indice(a,b,n,x);        // Calcul de l'indice tel que x_i <= x < x_{i+1}
   xi = a+(i-1)*h;           // et de la valeur correspondante de x_i
   xip= a+    i*h;           // et de x_{i+1}

   zgd=(Y(i+1)-Y(i))/h;      // La valeur de z_{g/d} du corrigé

   // On utilise la formule donnée dans le corrigé

   val_pi =    Y(i+1)  *  (x-xi)/h ...
            - Y(i)     *  (x-xip)/h ...
            + (x-xi)  *  (x-xip)/h^2 ...
               *  (    (Z(i+1)-zgd)  *  (x-xi)  ...
                   +  (Z(i)-zgd)     *  (x-xip)   );

endfunction

function [val_pi_prime]= valeur_derivee_spline (a,b,n,Y,Z,x)

   h=(b-a)/(n-1);            // Pas de la discrétisation

   i=indice(a,b,n,x);        // Calcul de l'indice tel que x_i <= x < x_{i+1}
   xi = a+(i-1)*h;           // et de la valeur correspondante de x_i
   xip= a+    i*h;           // et de x_{i+1}

   zgd=(Y(i+1)-Y(i))/h;      // La valeur de z_{g/d} du corrigé

   // On utilise la formule donnée dans le corrigé

   val_pi_prime = zgd ...
         + (x-xi)^2/h^2   *  ( Z(i+1)-zgd ) ...
         + (x-xip)^2/h^2  *  ( Z(i)-zgd )   ...
         + 2*(x-xi)*(x-xip)/h^2  *  ( Z(i)+Z(i+1)-2*zgd);

endfunction
```

5. Encore une fois on utilise les fonctions précédemment définies pour estimer l'erreur pour chaque point tiré aléatoirement.

```
function [erreur]= erreur_spline (a,b,f,f_prime,n,M)

   // On calcule les coefficients de la spline

   [Y,Z] = parametres_spline (a,b,n,f,f_prime);

   // On tire les points aléatoires

   xbar=a+(b-a)*rand([1:M]);

   erreur=0;  // Initialisation de l'erreur

   // On calcule le max des erreurs sur l'echantillon

   for k=1:M
      erreur = max(erreur, ...
```

```
                    abs(   f(xbar(k))  ...
                         - valeur_spline(a,b,n,Y,Z,xbar(k))));
   end;

endfunction

function [erreur]= erreur_derivee_spline (a,b,f,f_prime,n,M)

   // On calcule les coefficients de la spline

   [Y,Z] = parametres_spline (a,b,n,f,f_prime);

   // On tire les points aléatoires

   xbar=a+(b-a)*rand([1:M]);

   erreur=0;  // Initialisation de l'erreur

   // On calcule le max des erreurs sur l'echantillon

   for k=1:M
      erreur = max(erreur, ...
                    abs(   fprime(xbar(k)) ...
                         - valeur_derivee_spline(a,b,n,Y,Z,xbar(k))));
   end;

endfunction
```

Pour faire les tracés demandés on reprend exactement le code de la question 2, la seule chose qui change étant la formule de l'erreur qu'on évalue et le fait qu'on a besoin de définir un nombre M de tirages aléatoires.
Voici le code pour vérifier que l'erreur $\|f-\pi\|_\infty$ est d'ordre 4.

```
a=0;
b=1;
deff('y=f(x)','y=sin(10*x+1)');
deff('y=f_prime(x)','y=10*cos(10*x+1)');

M=5000;  // Nombre de tirages aléatoires

erreurs=[];              // Initialisation d'un tableau vide

for n=10:20:200          // Boucle sur le nombre de pas
  x=linspace(a,b,n)';    // En colonne
  [Y,Z]= parametres_spline (a,b,n,f,f_prime);    // Calcul de la spline
  erreurs=[erreurs; ...
           (b-a)/(n-1)  erreur_spline(a,b,f,f_prime,n,M)];
end;

// On trace l'erreur en fonction du pas
plot(erreurs(:,1), erreurs(:,2),'-*');

// On trace h --> h^4 en rouge pour comparer les pentes
plot(erreurs(:,1), erreurs(:,1).^4,'r');

h=gca();
h.log_flags='lln';  // Echelle logarithmique
```

Et celui pour vérifier que l'erreur $\|f'-\pi'\|_\infty$ est d'ordre 3.

```
a=0;
b=1;
deff('y=f(x)','y=sin(10*x+1)');
deff('y=f_prime(x)','y=10*cos(10*x+1)');

M=5000;  // Nombre de tirages aléatoires

erreurs=[];              // Initialisation d'un tableau vide

for n=10:20:200          // Boucle sur le nombre de pas
  x=linspace(a,b,n)';    // En colonne
  [Y,Z]= parametres_spline (a,b,n,f,f_prime);    // Calcul de la spline
  erreurs=[erreurs; ...
```

```
            (b-a)/(n-1)  erreur_derivee_spline(a,b,f,f_prime,n,M)];
end;

// On trace l'erreur en fonction du pas
plot(erreurs(:,1), erreurs(:,2),'-*');

// On trace h --> h^3 en rouge pour comparer les pentes
plot(erreurs(:,1), erreurs(:,1).^3,'r');

h=gca();
h.log_flags='lln';  // Echelle logarithmique
```